JEST: N+1-version Differential Testing of Both JavaScript Engines and Specification

Jihyeok Park, Seungmin An, Donjun Youn, Geyongwon Kim, Sukyoung Ryu

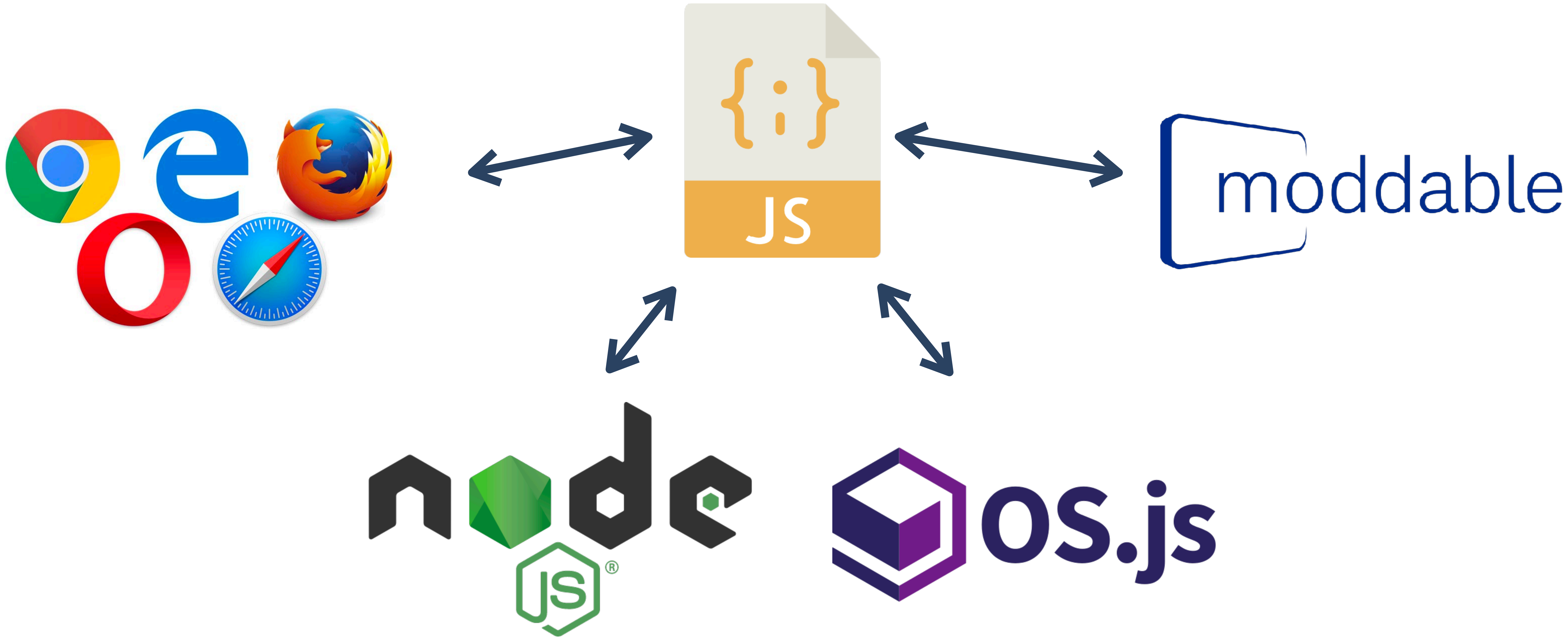
PLRG @ KAIST

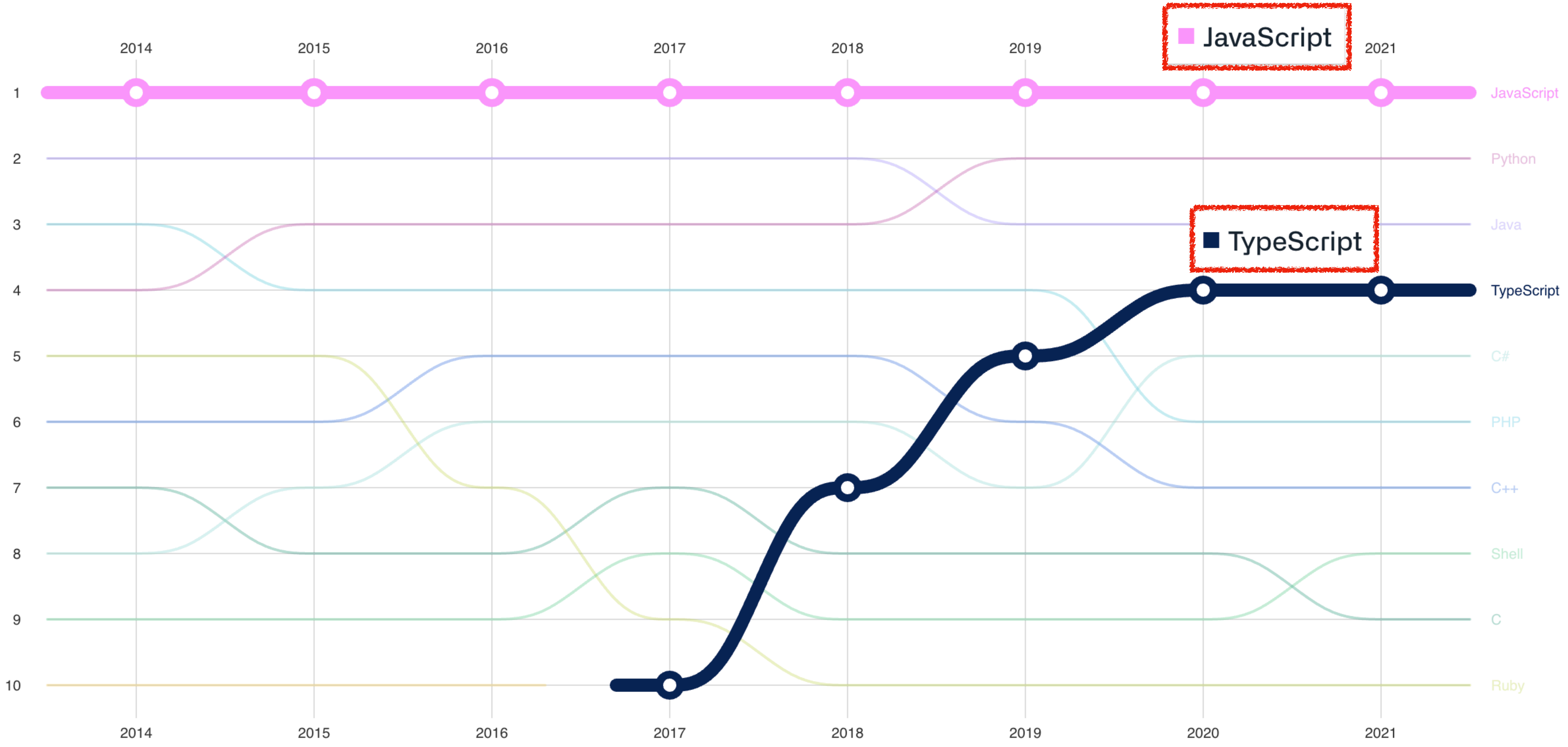
The 43rd International Conference on Software Engineering **(ICSE 2021)**
(Awarded ACM SIGSOFT Distinguished Paper)

2022 한국 소프트웨어공학 학술대회 (KCSE 2022) 초청 논문 발표

January 20, 2022

JavaScript is Everywhere





https://octoverse.github.com/

JavaScript Complex Semantics

```
function f(x) { return x == !x; }
```

Always return `false`?

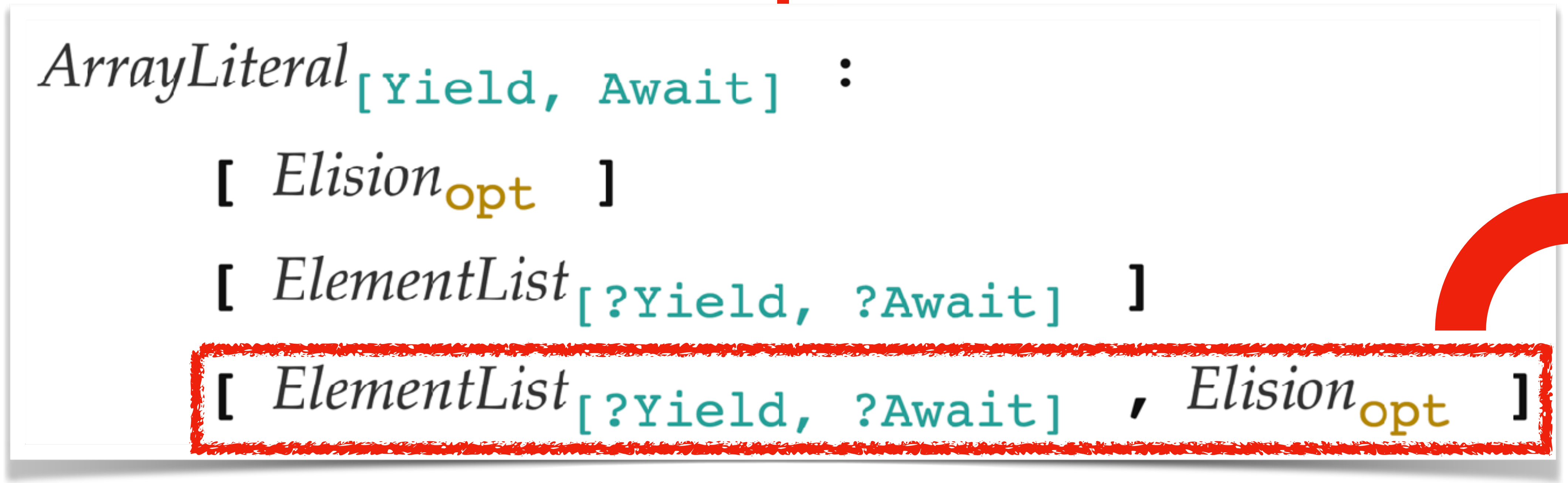
NO!!

```
f([]) -> [] == ![]
      -> [] == false
      -> +[] == +false
      -> 0 == 0
      -> true
```

ECMAScript: JavaScript Specification

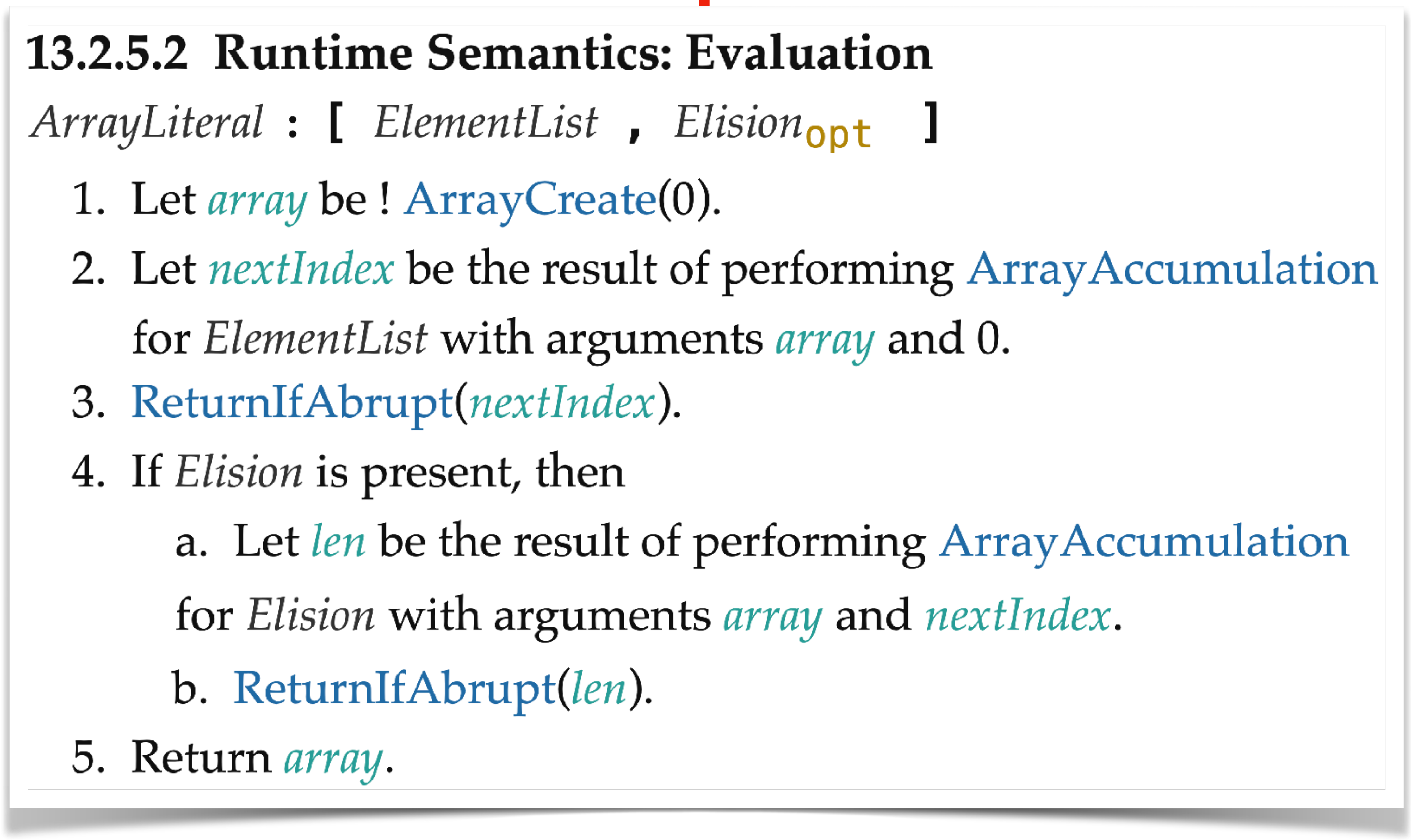
Syntax

Semantics



ArrayLiteral$_{\texttt{[Yield, Await]}}$:
- `[` *Elision*$_{\texttt{opt}}$ `]`
- `[` *ElementList*$_{\texttt{[?Yield, ?Await]}}$ `]`
- `[` *ElementList*$_{\texttt{[?Yield, ?Await]}}$ `,` *Elision*$_{\texttt{opt}}$ `]`

The production of *ArrayLiteral* in ES12



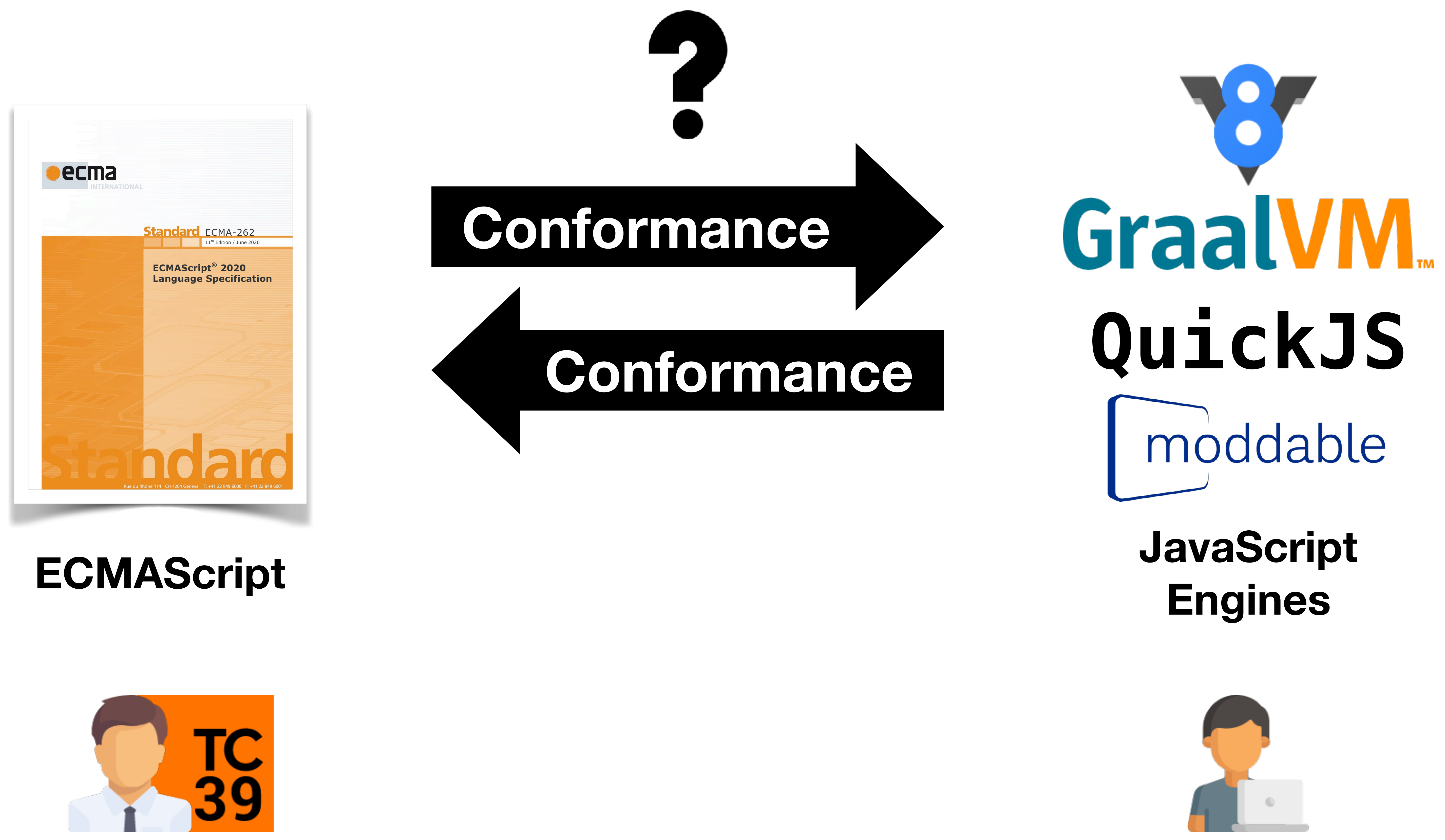
13.2.5.2 Runtime Semantics: Evaluation

ArrayLiteral : `[` *ElementList* `,` *Elision*$_{\texttt{opt}}$ `]`

```
1. Let array be ! ArrayCreate(0).
2. Let nextIndex be the result of performing ArrayAccumulation for ElementList with arguments array and 0.
3. ReturnIfAbrupt(nextIndex).
4. If Elision is present, then
    a. Let len be the result of performing ArrayAccumulation for Elision with arguments array and nextIndex.
    b. ReturnIfAbrupt(len).
5. Return array.
```

The Evaluation algorithm for
the third alternative of *ArrayLiteral* in ES12

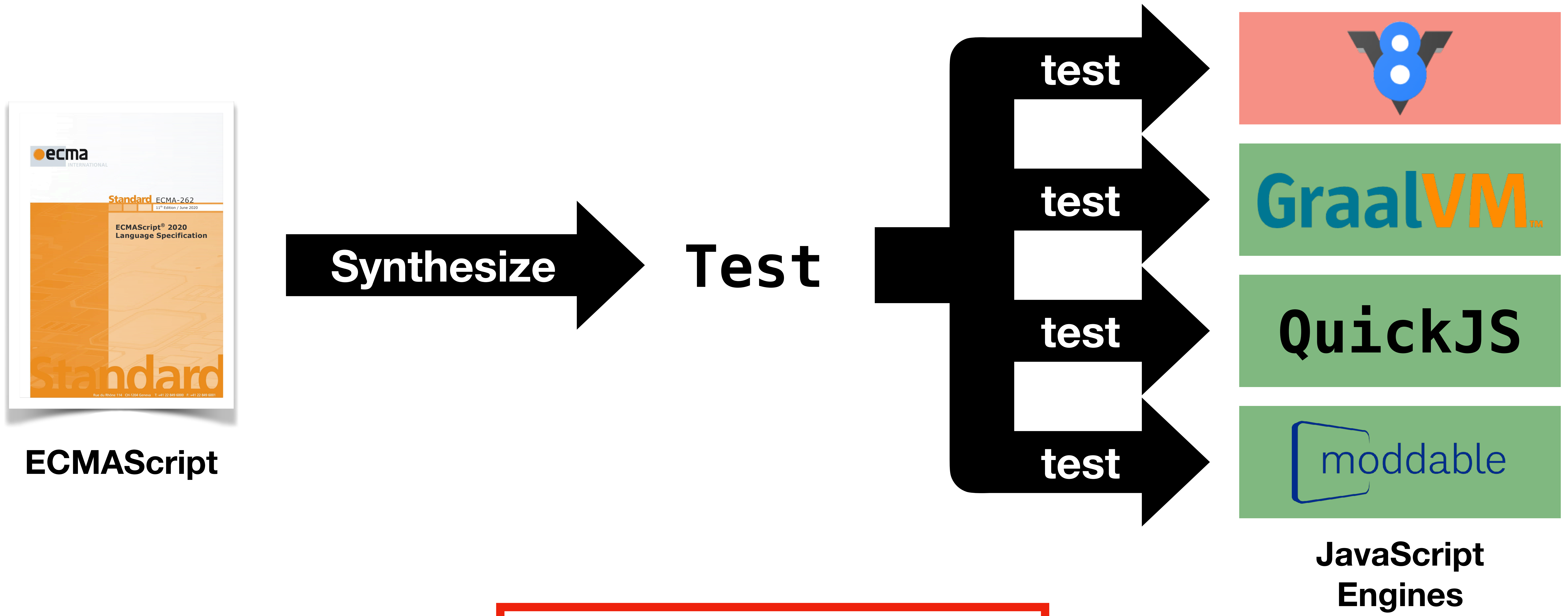
JavaScript Specification and Engines



ECMAScript

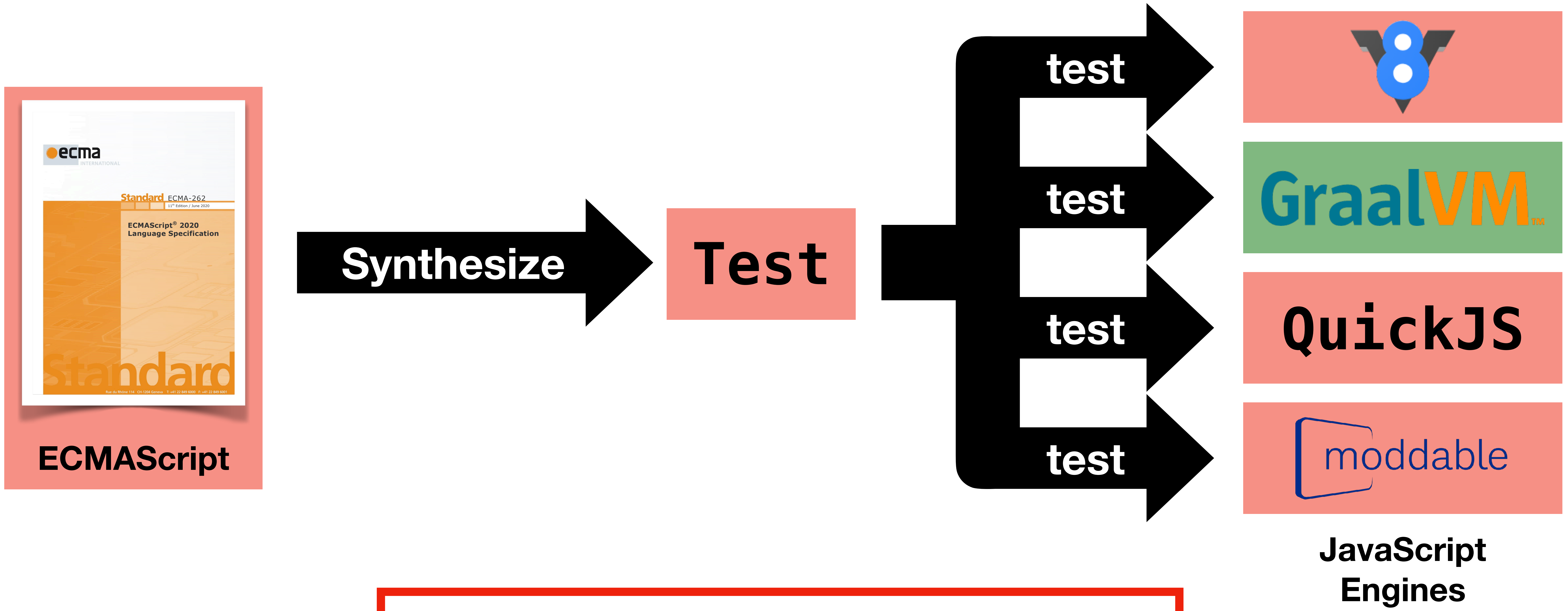
JavaScript Engines

Our Idea: N+1-version Differential Testing



An engine bug in

Our Idea: N+1-version Differential Testing

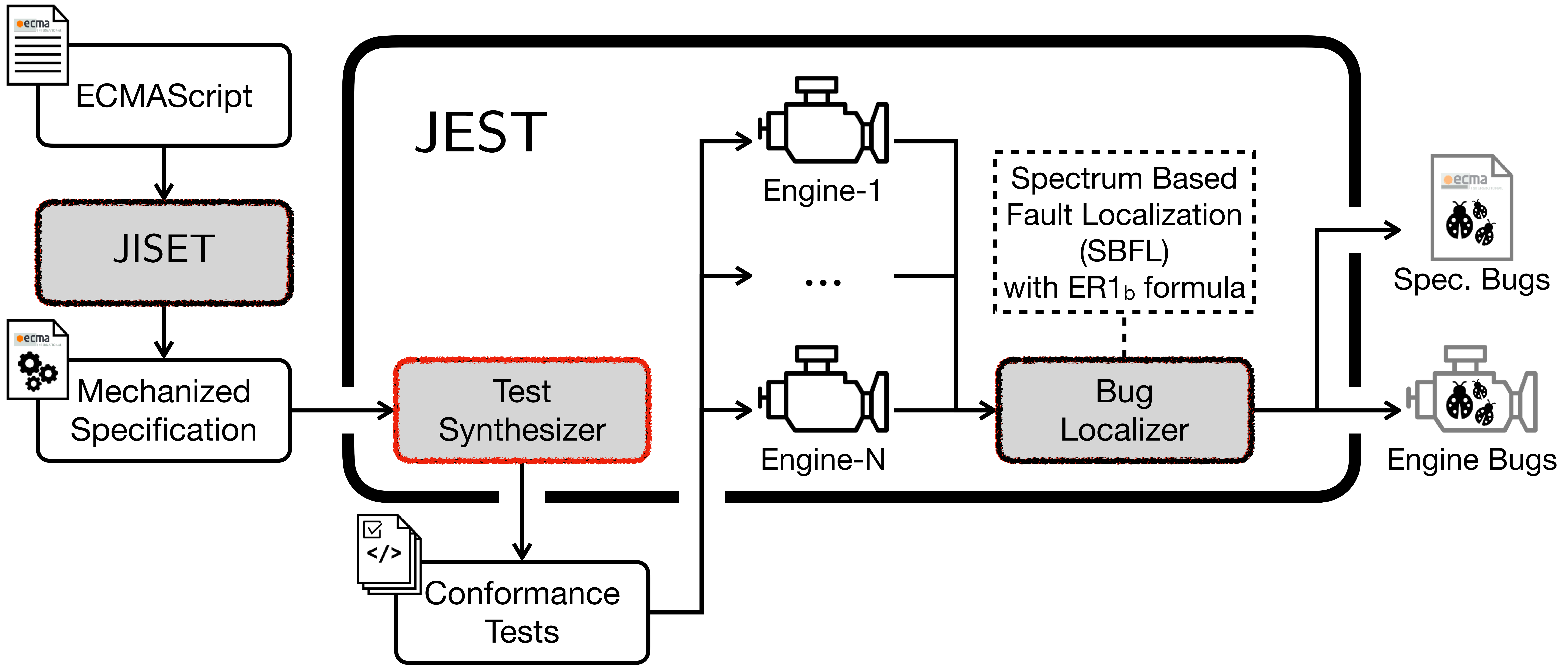


A specification bug in ECMAScript

An engine bug in GraalVM

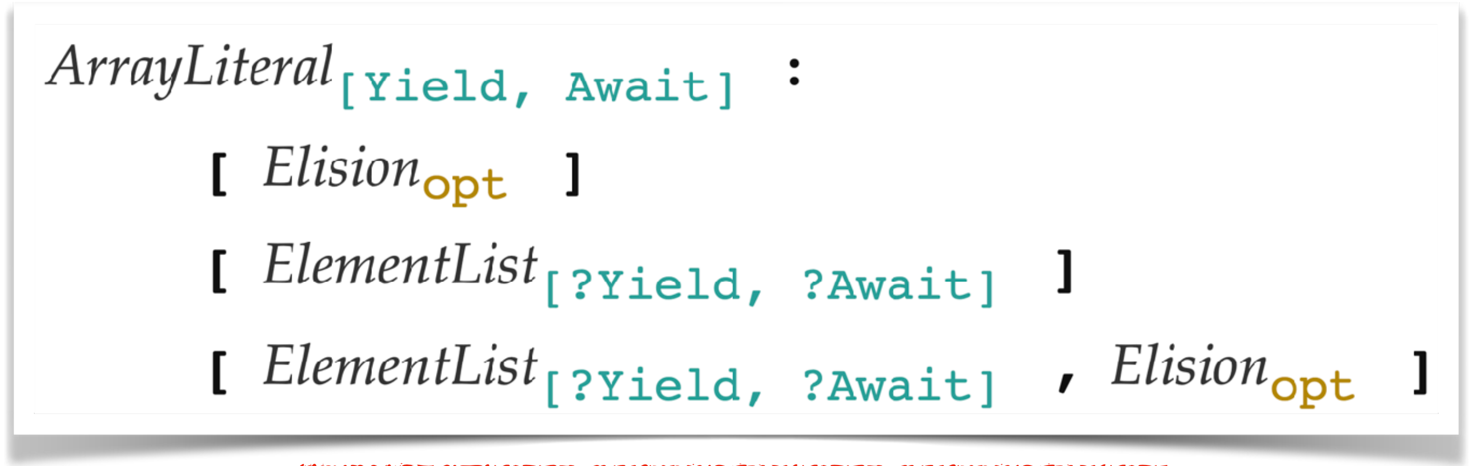
JEST

JavaScript **E**ngines and **S**pecification **T**ester



[ASE'20] Park et al, "JISET: Javascript IR-based Semantics Extraction Toolchain"

JEST - Test Synthesizer

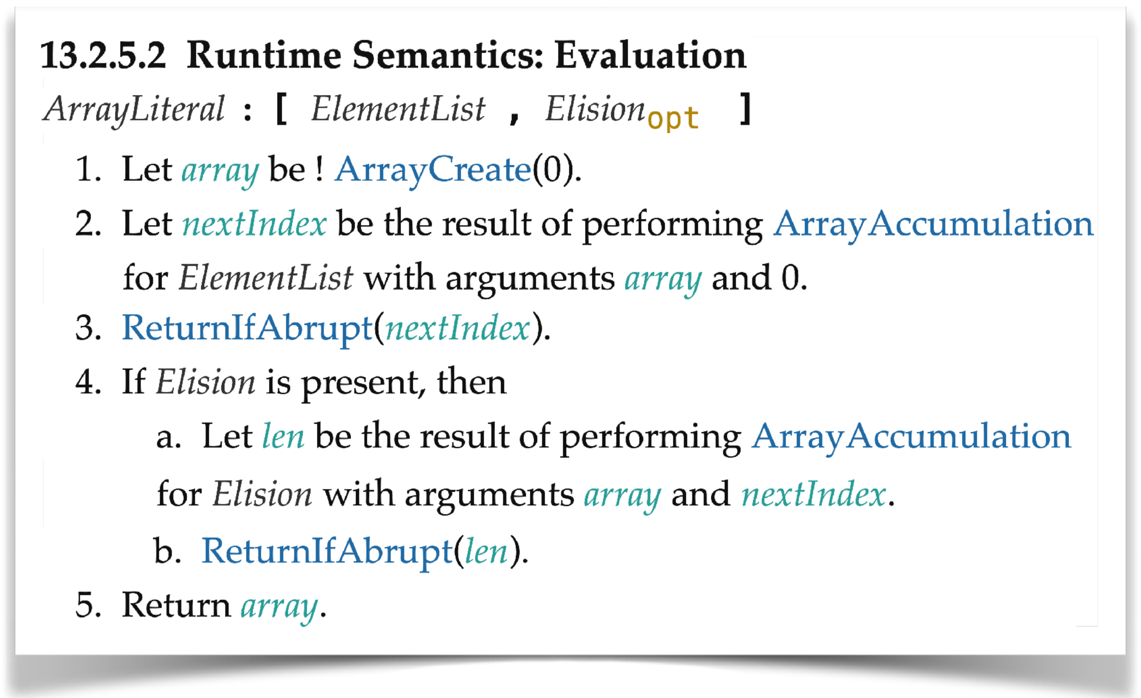


$ArrayLiteral_{\texttt{[Yield, Await]}}$:

[$Elision_{\texttt{opt}}$]

[$ElementList_{\texttt{[?Yield, ?Await]}}$]

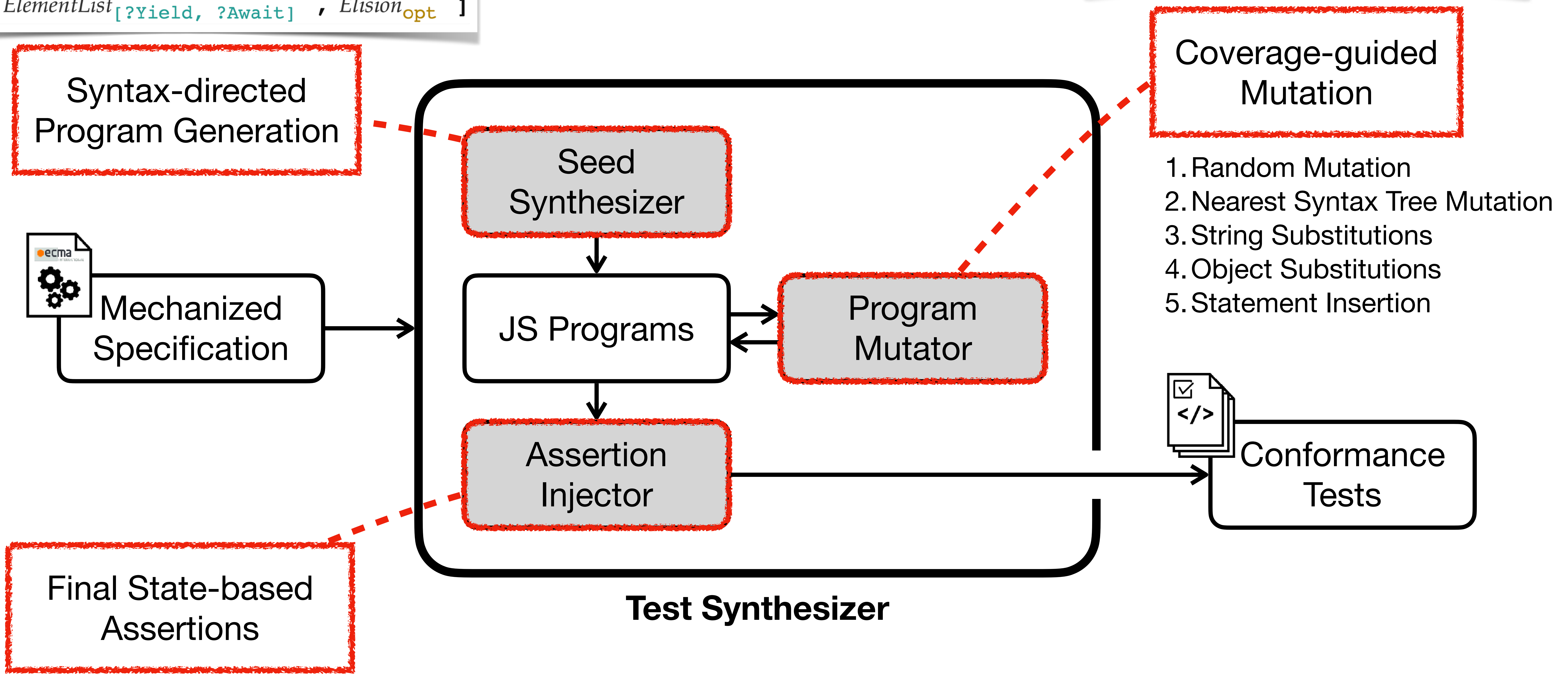
[$ElementList_{\texttt{[?Yield, ?Await]}}$, $Elision_{\texttt{opt}}$]



13.2.5.2 Runtime Semantics: Evaluation

$ArrayLiteral$: [$ElementList$, $Elision_{\texttt{opt}}$]

1. Let *array* be ! ArrayCreate(0).
2. Let *nextIndex* be the result of performing ArrayAccumulation for *ElementList* with arguments *array* and 0.
3. ReturnIfAbrupt(*nextIndex*).
4. If *Elision* is present, then
   a. Let *len* be the result of performing ArrayAccumulation for *Elision* with arguments *array* and *nextIndex*.
   b. ReturnIfAbrupt(*len*).
5. Return *array*.



1. Random Mutation
2. Nearest Syntax Tree Mutation
3. String Substitutions
4. Object Substitutions
5. Statement Insertion

JEST - Assertion Injector (7 Kinds)

1. **Variable Values (`Var`)**

```
  var x = 1 + 2;
+ $assert.sameValue(x, 3);
```

2. **Object Values (`Obj`)**

```
  var x = {}, y = {}, z = { p: x, q: y };
+ $assert.sameValue(z.p, x);
+ $assert.sameValue(z.q, y);
```

3. **Exceptions (`Exc`)**

```
+ // Throw (SyntaxError)
  let x = 42;
  function x() {};
```

4. **Aborts (`Abort`)**

```
+ // Abort
  var x = 42; x++;
```

JEST - Assertion Injector (7 Kinds)

5. **Object Properties** (Desc)

```
  var x = { p: 42 };
+ $verifyProperty(x, "p", {
+   value: 42.0, writable: true,
+   enumerable: true, configurable: true
+ });
```

6. **Property Keys** (Key)

```
  var x = {[Symbol.match]: 0, p: 0, 3: 0, q: 0, 1: 0}
+ $assert.compareArray(
+   Reflect.ownKeys(x),
+   ["1", "3", "p", "q", Symbol.match]
+ );
```

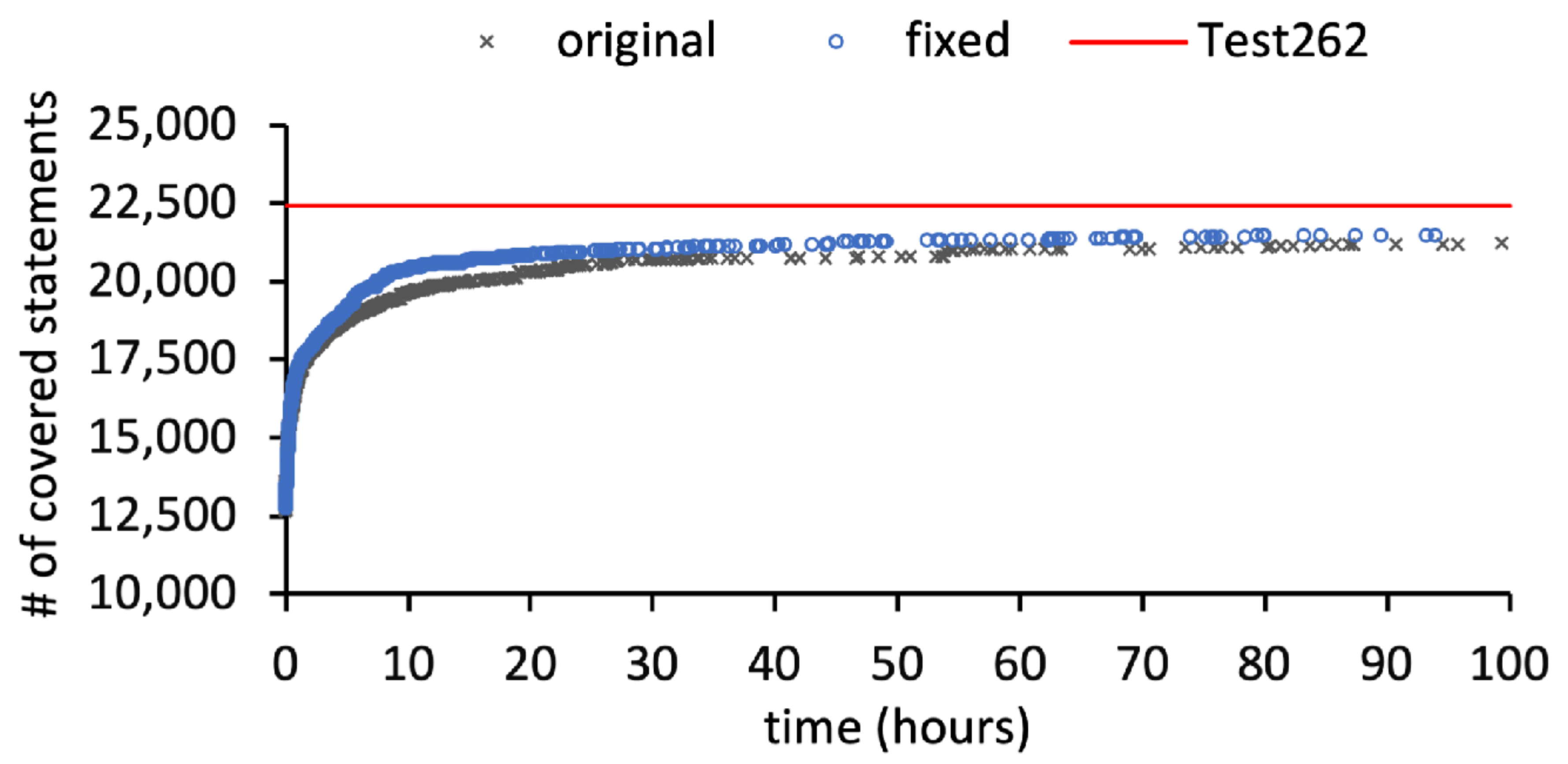
7. **Internal Methods and Slots** (In)

```
  function f() {}
+ $assert.sameValue(Object.getPrototypeOf(f),
+                   Function.prototype);
+ $assert.sameValue(Object.isExtensible(x), true);
+ $assert.callable(f);
+ $assert.constructable(f);
```

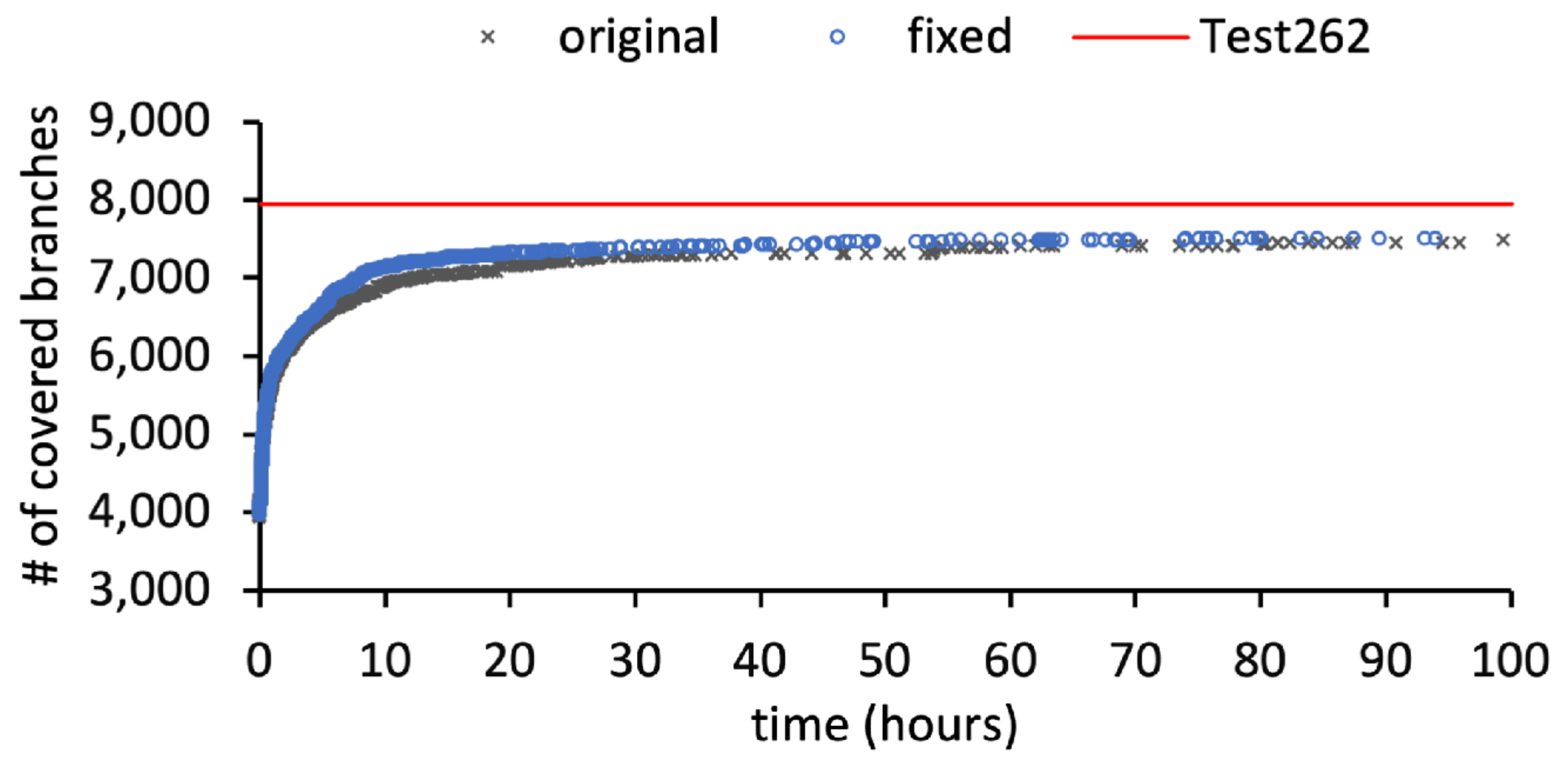
Evaluation

- **JavaScript Specification**
  - ECMAScript 2020 (ES11) - released in June 2020
- **JavaScript Engines**
  - **V8** - v8.3 by Google
  - **GraalJS** - v20.1.0 by Oracle
  - **QuickJS** - 2020-04-12 by Fabrice Bellard
  - **Moddable XS** - v10.3.0 by Moddable Tech Inc.

RQ1: Coverage of Synthesized Tests



(a) Statement coverage



(b) Branch coverage

- 1,700 **Synthesized Tests in** 100 hours
- **Syntax Coverage:** 97.79% (397 / 406)
- **Semantics Coverage**
  - Statement: 86.67% (21,230 / 24,495)
  - Branch: 77.95% (7,480 / 9,596)

RQ2: Bug Detection in JavaScript Engines

TABLE II: The number of engine bugs detected by JEST

Engines	Exc	Abort	Var	Obj	Desc	Key	In	Total
V8	0	0	0	0	0	2	0	2
GraalJS	6	0	0	0	2	8	0	16
QuickJS	3	0	1	0	0	2	0	6
Moddable XS	12	0	0	0	3	5	0	20
Total	21	0	1	0	5	17	0	44

44 Bugs in Engines

```
function f (... { x = x }) { return x; } var y = f();
```

QuickJS initializes '`x`' with '`undefined`' instead of throwing a '`ReferenceError`'

```
try { ++undefined; } catch(e) { }
```

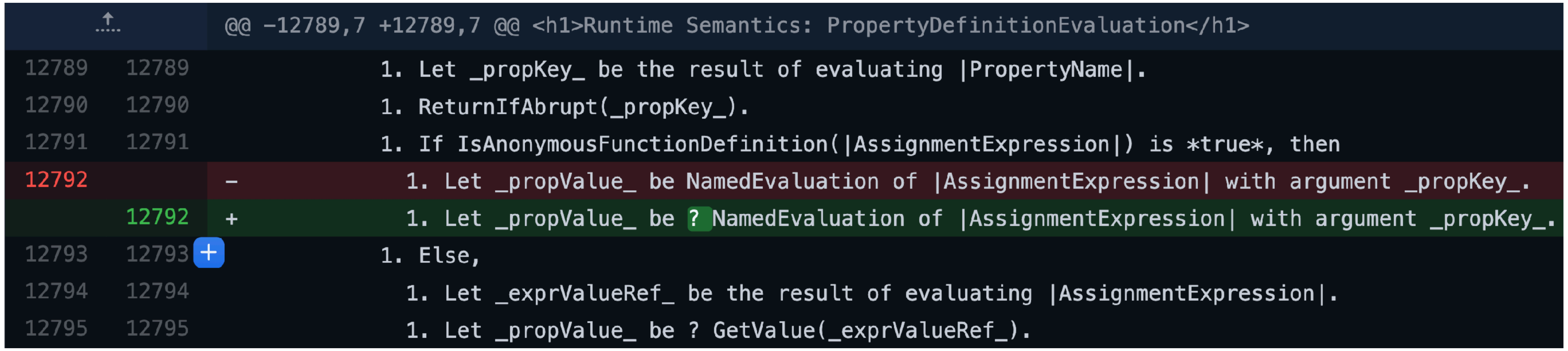
GraalJS crashes with an exception '`java.lang.IllegalStateException`'

RQ3: Bug Detection in ECMAScript

27 Bugs in Spec.

TABLE III: Specification bugs in ECMAScript 2020 (ES11) detected by JEST

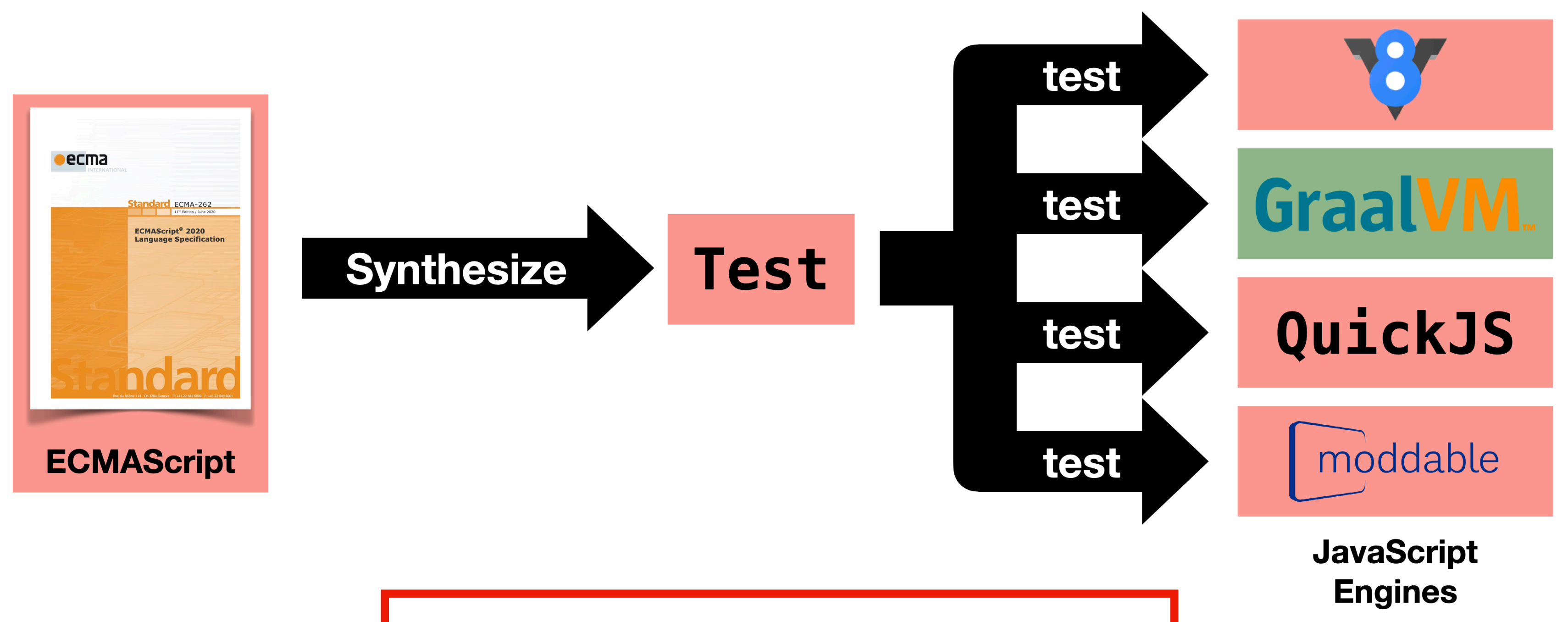
Name	Feature	#	Assertion	Known	Created	Resolved	Existed
ES11-1	Function	12	Key	O	2019-02-07	2020-04-11	429 days
ES11-2	Function	8	Key	O	2015-06-01	2020-04-11	1,776 days
ES11-3	Loop	1	Exc	O	2017-10-17	2020-04-30	926 days
ES11-4	Expression	4	Abort	O	2019-09-27	2020-04-23	209 days
ES11-5	Expression	1	Exc	O	2015-06-01	2020-04-28	1,793 days
ES11-6	Object	1	Exc	X	2019-02-07	2020-11-05	637 days



```
@@ -12789,7 +12789,7 @@ <h1>Runtime Semantics: PropertyDefinitionEvaluation</h1>
          1. Let _propKey_ be the result of evaluating |PropertyName|.
          1. ReturnIfAbrupt(_propKey_).
          1. If IsAnonymousFunctionDefinition(|AssignmentExpression|) is *true*, then
-           1. Let _propValue_ be NamedEvaluation of |AssignmentExpression| with argument _propKey_.
+           1. Let _propValue_ be ? NamedEvaluation of |AssignmentExpression| with argument _propKey_.
          1. Else,
            1. Let _exprValueRef_ be the result of evaluating |AssignmentExpression|.
            1. Let _propValue_ be ? GetValue(_exprValueRef_).
```

https://github.com/tc39/ecma262/pull/2130/files

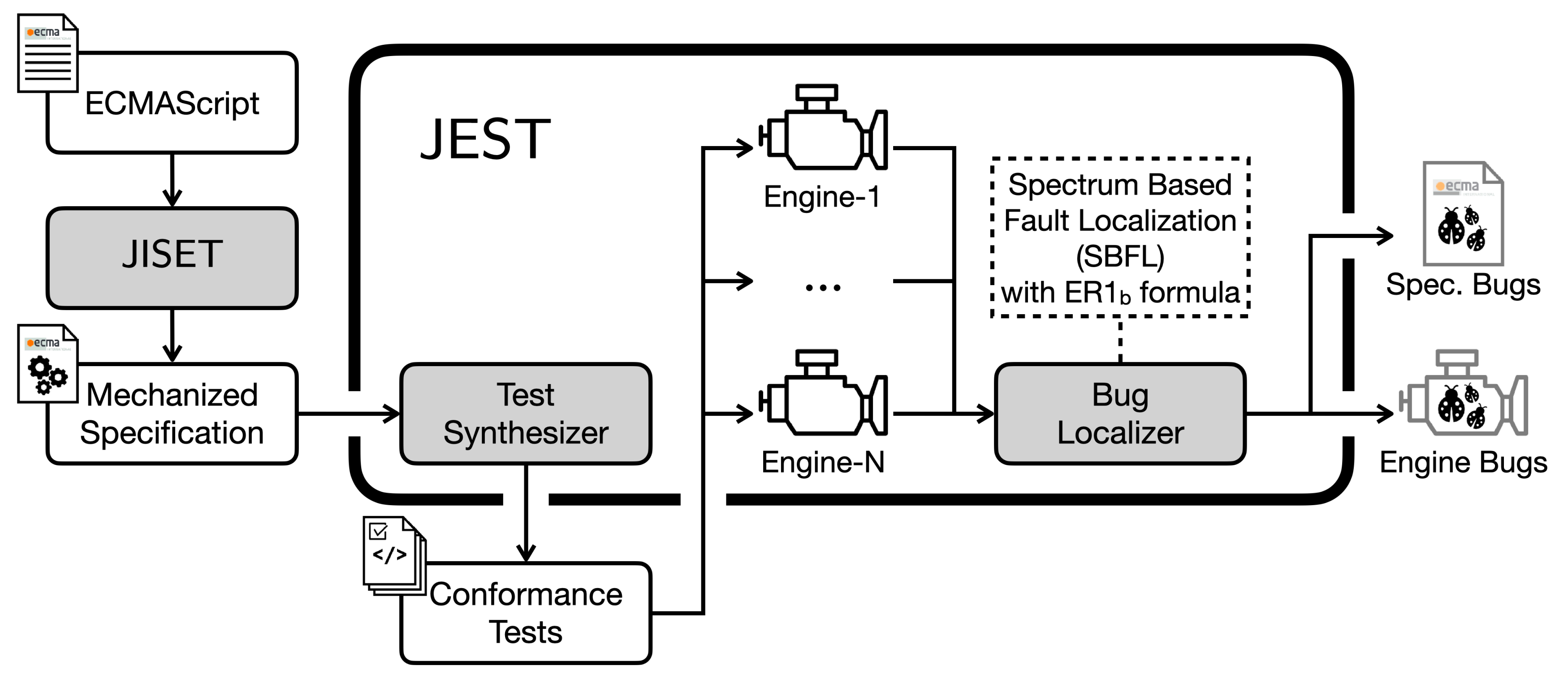
Our Idea: N+1-version Differential Testing



A **specification** bug in ECMAScript
An **engine** bug in GraalVM

JEST

JavaScript Engines and Specification Tester



[ASE'20] Park et al, "JISET: Javascript IR-based Semantics Extraction Toolchain"

RQ2: Bug Detection in JavaScript Engines

TABLE II: The number of engine bugs detected by JEST

Engines	Exc	Abort	Var	Obj	Desc	Key	In	Total
V8	0	0	0	0	0	2	0	2
GraalJS	6	0	0	0	2	8	0	16
QuickJS	3	0	1	0	0	2	0	6
Moddable XS	12	0	0	0	3	5	0	20
Total	21	0	1	0	5	17	0	44

44 Bugs in Engines

```
function f (... { x = x }) { return x; } var y = f();
```

QuickJS initializes 'x' with '`undefined`' instead of throwing a '`ReferenceError`'

```
try { ++undefined; } catch(e) { }
```

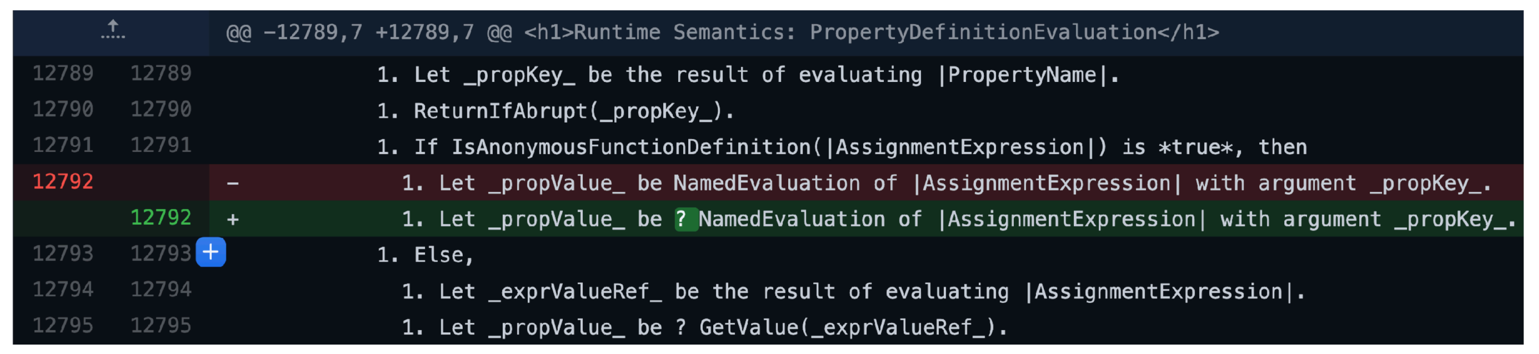
GraalJS crashes with an exception '`java.lang.IllegalStateException`'

RQ3: Bug Detection in ECMAScript

27 Bugs in Spec.

TABLE III: Specification bugs in ECMAScript 2020 (ES11) detected by JEST

Name	Feature	#	Assertion	Known	Created	Resolved	Existed
ES11-1	Function	12	Key	O	2019-02-07	2020-04-11	429 days
ES11-2	Function	8	Key	O	2015-06-01	2020-04-11	1,776 days
ES11-3	Loop	1	Exc	O	2017-10-17	2020-04-30	926 days
ES11-4	Expression	4	Abort	O	2019-09-27	2020-04-23	209 days
ES11-5	Expression	1	Exc	O	2015-06-01	2020-04-28	1,793 days
ES11-6	Object	1	Exc	X	2019-02-07	2020-11-05	637 days



```
@@ -12789,7 +12789,7 @@ <h1>Runtime Semantics: PropertyDefinitionEvaluation</h1>
            1. Let _propKey_ be the result of evaluating |PropertyName|.
            1. ReturnIfAbrupt(_propKey_).
            1. If IsAnonymousFunctionDefinition(|AssignmentExpression|) is *true*, then
 -            1. Let _propValue_ be NamedEvaluation of |AssignmentExpression| with argument _propKey_.
 +            1. Let _propValue_ be ? NamedEvaluation of |AssignmentExpression| with argument _propKey_.
            1. Else,
              1. Let _exprValueRef_ be the result of evaluating |AssignmentExpression|.
              1. Let _propValue_ be ? GetValue(_exprValueRef_).
```

https://github.com/tc39/ecma262/pull/2130/files